מחשבת

גרפיקת השחקנים

ATARI 7

600XL 800XL ATARI 400 and 800

ATARI
600XL, 800XL, 400, 800

מחשבת

כלים לפיתוח משחקי ארקייד
בלימוד עצמי

יחידה 7

גרפיקת השחקנים
(PLAYER MISSILE GRAPHICS)

תוכן העניינים

הקדמת המחבר

בסיומה של מחשבת 6, אשר יצאה לאור בשנת 1986 הבטיחו מחברי הסדרה כי מחשבת 7 תעסוק בלימוד ה-"PLAYER MISSILE" המיוחדים של מחשב האטארי. חלפו למעלה משלושים שנה של ציפייה דרוכה, ומחשבת 7 אין! אשר על כן, החלטתי להרים את הכפפה ולהשלים את המלאכה בעצמי.

עולם המחשוב בן-ימינו שונה לאין ערוך מזה של שנות השמונים, וכשניגשתי אל מלאכת הכתיבה ביקשתי בעצם לחזור אחורה בזמן – אימצתי את סגנון הכתיבה של ששת יחידות הלימוד הקיימות, נצמדתי לאותו מבנה של משימות ופתרונות, והעיקר – הקפדתי לכפות על עצמי את הגבלות החומרה והתוכנה שהיו נגישים אז – קרי: מחשב אטארי 800XL וגרסת הבייסיק הבסיסית שהותקנה עליו.

מיהו קהל היעד שלי? ובכן, הלוואי שהייתה לי מכונת זמן ויכולתי לשלוח את יחידת הלימוד הזו אל נדב הצעיר שבילה את ימי נעוריו בתכנות בבייסיק על מחשב האטארי שלו. אך מכיוון שעוד אין בנמצא מכונת זמן שכזו, הרי שהחוברת מיועדת לכל מי שמחשבי הרטרו יקרים ללבו, וששפת התכנות בייסיק – עוד לא נס ליחה בעבורו. מי יתן ומפעל זה יעורר עניין בקרב ילדי שנות השמונים ואולי אף בקרב חובבי הרטרו מדור המילניום, ואם אעשה עוד כמה נפשות לאהבת התכנות בבייסיק, הרי שזה שכרי על כל עמלי.

אני מוקיר תודה עמוקה לידידי ניר דרעי, אשר הפגיש אותי עם תחביב ישן, השיב לי את האהבה למחשב ילדותי, וגם השיא לי עצות רבות וטובות במהלך כתיבתו והפקתו של ספר זה.

נדב שטכמן פולישוק
ינואר 2019

מבוא

ודאי תופתע לגלות כי במחשב האטארי שלך יש מאפיין ייחודי שנקרא "גרפיקת שחקנים" או PLAYER MISSILE GRAPHICS, ובקיצור: PMG.

בעזרת אותה טכנולוגיה ייחודית תוכל ליצור ולעצב אלמנטים גרפיים מרהיבים, לצבוע אותם בכל צבע שתחפוץ ולהניע אותם בקלות רבה על פני מסך הטלביזיה שלך.

דע לך כי בתוך המחשב שלך מצויים שני מיני-מעבדים ייחודיים (בנוסף ל"מוח" המרכזי) הנקראים ANTIC ו-GTIA. מיני-מעבדים אלו שולטים בגרפיקת השחקנים ולמעשה מאפשרים לך להשתמש באותם אלמנטים גרפיים ייחודיים.

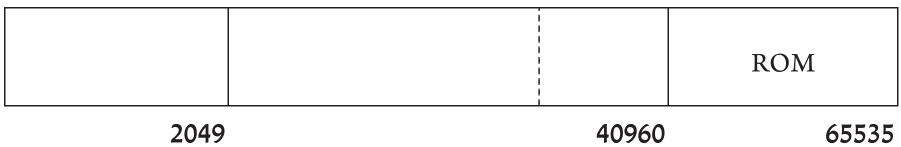
בשלב זה ייתכן שאתה מעט מבולבל, אך אל דאגה, כרגע רק הצגנו לפניך את הטכנולוגיה. בהמשך החוברת נעבור איתך שלב אחר שלב עד שתדע ליצור את האלמנטים הללו בעצמך.

כשתגיע לסיומה של יחידת לימוד זו תוכל לפתח בכוחות עצמך משחקי פעולה מעניינים ומרתקים שלא נופלים ברמתם ממשחקי המחשב ומשחקי הארקייד המוכרים לך.

ובכן, הבה נצא לדרך!

פרק א
מושגי יסוד בגרפיקת השחקנים

בפרק א' של מחשבת 6 "מסע אל זכרונו של המחשב" הצגנו לפניך באופן גרפי את מבנה ה-RAM וה-ROM של המחשב.



אתה כבר יודע כי בבתים ממספר 40960 ועד 65535 נמצאים ערכים קבועים הניתנים לקריאה בלבד ואי אפשר לשנותם.

? מה לדעתך יקרה אם תפקוד על המחשב את שתי הפקודות הבאות?

```
POKE 53277, 3
POKE 53248, 48
```

לכאורה לא אמור לקרות דבר, משום שבתים אלו נמצאים עמוק בתוך אזור ה-ROM של הזיכרון.

הפתעה! בצדו הימני של מסך הטלביזיה שלך מופיע מעין צינור מים תזזיתי שמשתולל ללא הרף.

■ עתה פקוד על המחשב:

```
? PEEK (53248)
```

המחשב משיב: 0

? מה קרה כאן?

ערכו של הבית 53248 לא השתנה כמובן (משום שבית זה נמצא באזור ה-ROM), אך יחד עם זאת המחשב כן הגיב לפקודה שלך והפעיל את אותו צינור תזזיתי.

מבלי להיכנס לעובי הקורה של מה מתחולל בתוך קרבי המחשב שלך, נסביר רק שבתוך ה-ROM ישנם "טריגרים" – בתים המאפשרים לך לפקוד על המחשב לבצע פעולות מסוימות, אך הערך שאתה פוקד לתוכם משמש כ"צל חולף" וכהרף עין הם חוזרים לערכם המקורי.

■ לחץ כעת על מקש ה-RESET כדי להפסיק את אותה זרימה תזזיתית.

דע לך כי כתובת הזיכרון 53277 משמשת מעין "טריגר" או "מתג" המורה למחשב לפתוח לשימוש את גרפיקת השחקנים (ה-PMG).
כאשר הצבת (לרגע) אל תוך כתובת זיכרון זו את הערך 3, למעשה הורית למחשב לפתוח לשימוש את כל ה"שחקנים" וה"טילים" שלו.

תא הזיכרון 53248, מורה למחשב היכן למקם את השחקן הראשון (אותו אלמנט גרפי שיצרת) על הציר האופקי, כלומר לרוחב המסך.
כאשר הצבת – לרגע – אל תוך כתובת זיכרון זו את הערך 48, למעשה הורית למחשב למקם את האלמנט הגרפי שיצרת (אותו צינור תזזיתי) בצדו השמאלי של מסך הטלביזיה – בדיוק במקום בו נמצא הסמן.

ואולם כדי שנוכל לעשות שימוש יעיל בגרפיקת השחקנים ולעצב דמויות של ממש (ולא רק צינורות תזזיתיים) עלינו לבחון עוד כתובת זיכרון חשובה – 54279.
כתובת זו מורה למחשב מהיכן "לשאוב" את הנתונים עבור האלמנטים הגרפיים שתיצור.

במחשבת 6 למדת על דרך הפעולה של "בתים מצביעים" (POINTERS).
אם אינך זוכר, עיין בפרק ג' של מחשבת 6.

ממש לפי אותו עיקרון, גם כתובת הזיכרון 54279 מתנהגת כמו בית מצביע.
כך למשל, אם תפקוד על המחשב:

```
POKE 54279, 152
```

? על איזה בית בזיכרון יצביע המחשב? ________________

(תשובה 1 בעמ' 63)

נסכם את מה שלמדנו עד כה:
1. למדנו "להדליק" את "מתג" ה-PMG
2. למדנו להודיע למחשב מאיזו כתובת "לשאוב" את המידע הגרפי הנחוץ עבור ה-PMG

עכשיו נותר לנו להקצות אזור בזיכרון ולהכניס לתוכו את המידע הגרפי.

? איזה גודל זיכרון נדרש לנו כדי לאכסן בתוכו את המידע הגרפי של האלמנטים שלנו?

כדי להשיב על השאלה הזו, נסקור את "מַצֵּבֶת כוח האדם" של גרפיקת השחקנים.

דע לך כי למחשב שלך יש אפשרות להציג עד ארבעה שחקנים בו זמנית על המסך. נוסף על כך, לכל שחקן אפשר להתאים "טיל" (או "שחקן קטן"), וגם הם יכולים לנוע באופן בלתי תלוי על המסך.

להלן איור המציג את מפת הזיכרון לגרפיקת השחקנים.



לעת עתה נתייחס רק לצד השמאלי של הטבלה – הפרדה אופקית של שתי שורות. בפרק האחרון בחוברת זו נתייחס בקצרה גם להפרדה האופקית של שורה אחת.

כל אחד מארבעת השחקנים תופס מקום של 128 בתים. 128 בתים נוספים נדרשים עבור כל ארבעת הטילים גם יחד.

? לכן סך כל הבתים הנדרשים לאכסון המידע הגרפי עבור השחקנים והטילים הוא: ____________

(תשובה 2, בעמ' 63)

בנוסף, עלינו להשאיר שטח ריק (BUFFER) בן 384 בתים בתחילת המערך.

כלומר: בסך הכול נדרשים לנו 1024 בתים או תאי זיכרון.

? היכן נקצה את המקום הזה בזיכרון?

במחשבת 6 למדת להזיז את ה-RAMTOP למקום אחר בזיכרון. אז עשית זאת לצורך בניית מערך האותיות העבריות שעיצבת. כמובן שאין מניעה להשתמש באותו אזור כדי למקם את המידע על גרפיקת השחקנים.

אם אינך זוכר כיצד עושים זאת, עיין בפרק ה' במחשבת 6.

? איזה ערך עליך להכניס לבית 106 כדי להוריד את ה-RAMTOP ב-1024 בתים?

____________ (תשובה 3, בעמ' 63)

■ אל תשכח לפקוד על המחשב: **GR.0** כדי לעדכן את המיקום החדש של ה-RAMTOP

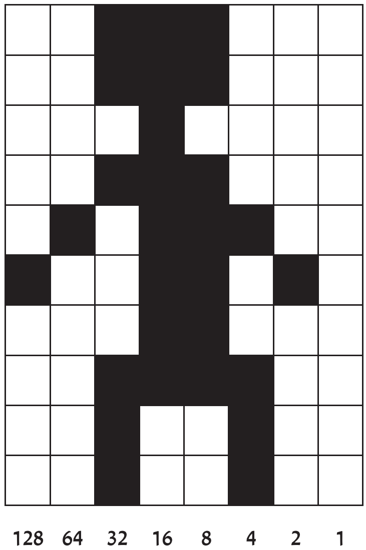
סוף כל סוף הגענו לשלב עיצוב האובייקט הגרפי הראשון שלך.

רוחבו של האלמנט הגרפי מוגבל ל-8 פיקסלים אך אורכו יכול להתפרש מן התחתית ועד לקצה גובה מסך הטלביזיה שלך.

אנו ממליצים לך לבנות אובייקטים קטנים יותר – למשל על מטריצה של 10X8.

גם כאן שיטת העיצוב והקידוד אינם חדשים לך והם זהים לשיטת העיצוב והקידוד של אותיות העברית שעיצבת במחשבת 6.

התבונן למשל באיש החביב הזה:



? נסה לקודד את ערכה של כל אחת מעשר השורות. לנוחיותך סימנו בתחתית המטריצה את ערכה של כל סיבית (BIT)
(אם אינך זוכר את שיטת החישוב, אתה מוזמן לעיין בפרק ד' שבמחשבת 6 ולהיזכר ב"שיטת המשקלים".)

____, ____, ____, ____, ____, ____, ____, ____, ____, ____

(תשובה 4, בעמ' 63)

בשלב זה אתה מוכן כמעט לגמרי לבנות את השחקן הראשון שלך ולהציבו על המסך.

נסכם את שלבי המשימה:

1 עליך להוריד את ה-RAMTOP ולהקצות 1024 תאים עבור זיכרון ה-PMG.
(אל תשכח את הפקודה: **GR. 0**)

2 עליך להורות למחשב היכן שמור המידע של גרפיקת השחקנים, כלומר להיכן עליו לגשת כדי "לשאוב" את המידע הגרפי.

3 עליך "לנקות" את השטח החדש שהקצית משאריות המידע המיותרים – כלומר לאפס את הערכים השמורים בבתים אלו.
(למעשה אינך חייב לאפס את כל 1024 הבתים, אלא רק את 128 הבתים המוקצים לשחקן)

4 עליך להכניס את נתוני הגרפיקה שחישבת אל המקום המתאים במערך הזיכרון.
(את המיקום המדויק תוכל לחשב בעזרת מפת השחקנים שהצגנו לעיל)

5 עליך "להדליק את המתג" של ה-PMG.

6 עליך להגדיר למחשב להשתמש ב"הפרדה אופקית של שורה אחת".
על מנת לעשות זאת, פנה לבית 559, ופקוד על המחשב:

```
POKE 559,46
```

7 עליך למקם את השחקן בנקודה האופקית הרצויה לך על המסך.

אנו מציעים לך לנסות להתמודד עם המשימה בכוחות עצמך, שלב אחר שלב. אם לא תצליח, התקדם לעמוד הבא.

נתקדם עתה שלב אחר שלב. לצורך הנוחיות נמספר את השורות בתוכנית בהתאם למספרי השלבים.

בשלב הראשון, עלינו להקצות 1024 בתים עבור זיכרון השחקנים. כדי להוריד את ה-RAMTOP ב-1024 בתים נמצא את ערכו של בית 106, ונפחית ממנו 4 (שכן לפי שיטת הפעולה של בתים מצביעים, 256X4 הם 1024 בתים).

```
10 A=PEEK(106)-4: POKE 106,A: GR.0
```

בשלב השני, נורה למחשב מהיכן לשאוב את נתוני ה-PMG.

```
20 POKE 54279,A
```

בשלבים הבאים, נכניס את נתוני ה-PMG שלנו לתוך אזור הזיכרון שהקצנו. על מנת לחשב את המקום המדויק שבו יושב המידע על השחקן הראשון, נתבונן בצד השמאלי של מפת השחקנים:

את תחילת זיכרון השחקנים, כלומר הבסיס, נכנה באותיות: PMBASE.

PMBASE	שטח ריק			
PMBASE+384	טיל 0	טיל 1	טיל 2	טיל 3
PMBASE+384+128	שחקן 0			
PMBASE+384+128+128	שחקן 1			
PMBASE+384+128+128+128	שחקן 2			
PMBASE+384+128+128+128+128	שחקן 3			
PMBASE+384+128+128+128+128+128				

השחקן הראשון שלנו, שחקן מספר 0, בנוי מ-128 בתים, הממוקמים החל מהמקום ה-512 מתחילת ה-PMBASE.
מדוע?
כזכור, 384 התאים הראשונים נותרים ללא שימוש, ו-128 התאים הבאים בתור, מוקצים לארבעת הטילים, ובסך הכול התקדמנו 512 תאים.

את נתוני גרפיקת השחקן הראשון (ערכיה של דמות האיש שחישבת קודם) נכניס במקום טוב באמצע: אם עומדים לרשותנו 128 בתים, נבחר למשל את התאים 50-59 ונכניס לתוכם את המידע הגרפי הרצוי לנו.
(שים לב כי הטווח של 50-59 הוא טווח של עשרה תאים, ולא תשעה!)

אבל לפני שנעשה זאת, חשוב ביותר לנקות את הטווח כולו – כלומר את כל 128 הבתים.

```
30 PMBASE=A*256: FOR R=PMBASE+512 TO
PMBASE+640: POKE R,0: NEXT R
34 FOR R=PMBASE+512+50 TO PMBASE+512+59:
READ M: POKE R,M: NEXT R
38 DATA 56,56,16,56,92,154,24,60,36,36
```

בשלב החמישי "נדליק את המתג" של ה-PMG:

```
40 POKE 53277,3
```

בשלב השישי, נגדיר למחשב להשתמש בהפרדה אופקית של שתי שורות:

```
50 POKE 559,46
```

ולבסוף, נמקם את האיש השחור שלנו במקום טוב לרוחב המסך. למשל על קואורדינטה 100:

```
60 POKE 53248, 100
```

צובעים את השחקן

אם לא לחצת על מקש RESET, השחקן שלך ניצב עדיין במרכז המסך, וצבעו שחור. (אם לחצת על המקש RESET, תצטרך להריץ את התוכנה מחדש)

■ פקוד עתה על המחשב:

```
POKE 704, 14
```

האיש שלך נצבע כולו בלבן.

דע לך כי בית 704 אחראי על צבעו של השחקן הראשון (שחקן מס' 0). תוכל לשחק עם הצבעים להנאתך ולהציב בתוך בית 704 כל ערך בין 0 ל-255.

איך תדע איזה ערך לפקוד לתוך בית 704 כדי לקבל את הצבע הרצוי לך?
ובכן, אתה כבר מכיר כמובן את הפקודה **SETCOLOR** המשמשת לקביעת צבעי המסך.
כך למשל, כדי לצבוע את צבע מסגרת המסך באדום, יש לפקוד על המחשב:

```
SETCOLOR 4, 3, 4
```

על מנת להגיע לאותו גוון אדום עבור השחקן שלך, נדרש חישוב פשוט:
את הערך הראשון של הצבע – 3 עליך להכפיל ב-16, ולתוצאה להוסיף את הערך השני – 4

? איזה ערך עליך להציב אם כן בתוך בית 704 כדי לקבל את הגוון האדום של המסגרת?

_______________ (תשובה 5, בעמ' 63)

אגב, בשלב הזה לא תופתע ודאי לשמוע כי גם צבעי המסך מאוכסנים בכמה בתי זיכרון ייעודיים. צבע מסגרת המסך למשל מאוכסן בבית 712 וגם כאן צורת החישוב זהה.
כך לדוגמה הפקודה:

```
SETCOLOR 4,12,8
```

שקולה לפקודה:

```
POKE 712,200
```

וזאת משום ש: (12X16)+8=200

כדי להשלים את התמונה הצבעונית, נספר לך כי בתים 708 עד 712 אחראים בהתאמה על "פחיות" צבע (רגיסטרים): 0, 1, 2, 3, 4

השחקן מתנועע על המסך

ראית כבר כי אפשר להניע את שחקן מס' 0 שמאלה וימינה בקלות רבה, באמצעות הבית: 53248.
כדי למקם את השחקן שלנו בקצה השמאלי של המסך, נפקוד את הערך 44.
כדי למקם את השחקן שלנו בקצה הימני של המסך, נפקוד את הערך 204.

? מה לדעתך יקרה אם נפקוד את הערך 0? ומה אם נפקוד את הערך 255?

נסה!

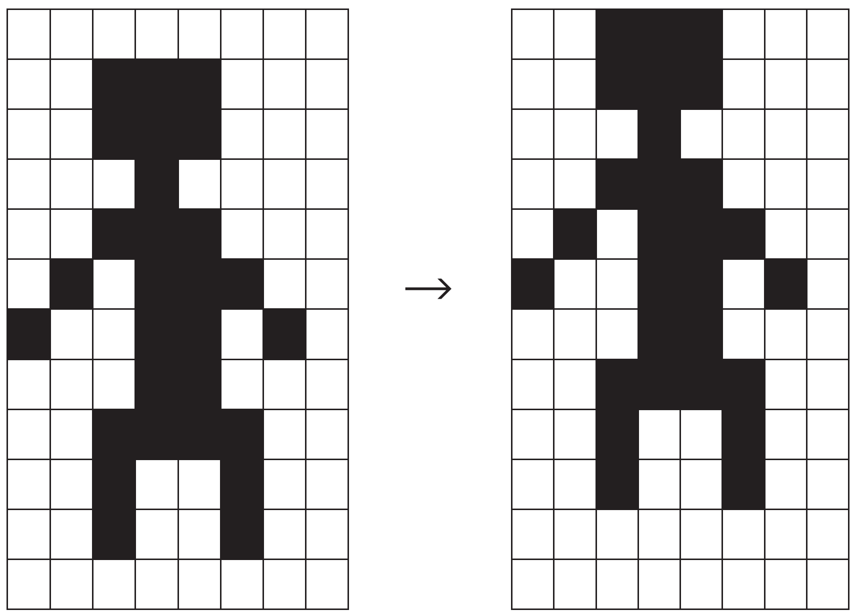
השחקן נעלם מן המסך. אך למעשה הוא מתחבא "מאחורי הקלעים".
POKE 53248,0 ממקם את השחקן בקצה השמאלי הבלתי נראה של המסך.
POKE 53248,255 ממקם את השחקן בקצה הימני הבלתי נראה של המסך.

טיפ: תוכל לפקוד את אחד מערכי קיצון אלו כדי להסתיר את השחקן שלך לזמן מה מהמסך.

אבל מה לגבי תנועה אנכית של השחקן – מלמעלה למטה (או להיפך).

כאן העניינים מתחילים להסתבך.

מכיוון שאין פקודה המזיזה את השחקן על הציר האנכי, אין ברירה אלא להכניס מחדש את כל נתוני השחקן למיקום חדש, כפי שמודגם בציור הבא:



■ נסה לכתוב בכוחות עצמך המשך לתוכנה, ובה השחקן שלנו (אותו איש חביב) מתנועע ברחבי המסך באמצעות הג'ויסטיק.

ראשית, כדי להניע את השחקן לרוחב המסך, כל שעליך לעשות הוא לקשר את תנועת הג'ויסטיק ימינה או שמאלה עם הבית האחראי על תנועתו האופקית של השחקן.

לגבי התנועה האנכית, פקוד על המחשב להזיז את ערכי הגרפיקה למיקומם החדש – הזזת הג'ויסטיק כלפי מעלה, תגביה את כל נתוני השחקן בשורה אחת ואילו הזזת הג'ויסטיק כלפי מטה, תנמיך את כל נתוני השחקן בשורה אחת.

רמז ראשון – אין צורך לקרוא מחדש (**READ–DATA**) את כל 10 נתוני הגרפיקה. יעיל יותר להורות למחשב לקרוא פעם אחר פעם את הערך המוצב בתא אחד ולהעתיק אותו לתא שמעליו (או מתחתיו).

רמז שני – אחרי שהעתקת את מיקומו של נתון הגרפיקה האחרון, אל תשכח לאפס את מיקומו הקודם (אחרת תקבל שחקן עם רגליים מתוחות או עם ראש הולך וגדל).

(תשובה 6, בעמ' 63)

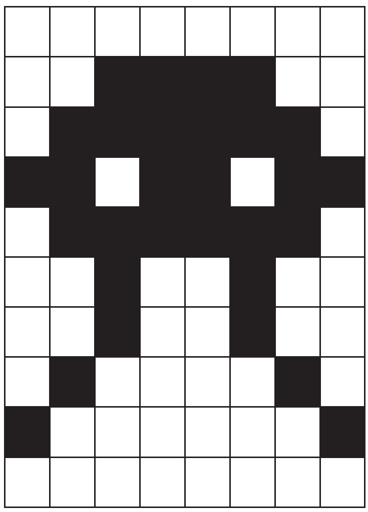
בוודאי שמת לב כי תנועת השחקן על הציר האופקי שונה באופייה מן התנועה על הציר האנכי. כשאתה מזיז את הג'ויסטיק ימינה או שמאלה, השחקן מתרוצץ בקלילות לרוחב המסך, אך כשאתה מזיז את הג'ויסטיק למעלה או למטה, תנועת השחקן איטית ומגושמת.

בפרקים הבאים של הספר נציע לך כמה דרכים להתמודד עם בעיה זו.

פרק ב
מוסיפים "שחקני חיזוק"

רוב משחקי המחשב כוללים יותר משחקן אחד. בחלק מן המשחקים על השחקן לחמוק משחקנים אחרים, לירות עליהם או לחמוק מירי נגדי.
על מנת שתוכל לתכנת את המשחק הראשון שלך, עליך ללמוד להשתמש בשחקנים נוספים, ובהמשך גם בטילים.

בשלב הראשון, נוסיף לשחקן שבנינו, שחקן חיזוק נוסף – מעין חייזר (שים לב שהשארנו שורה ריקה מעל ומתחת לחייזר כדי להזיזו למעלה, כפי שהסברנו קודם)



■ קודד את ערכה של כל שורה

____

(תשובה 7, בעמ' 64)

חזור עתה ועיין במפת גרפיקת השחקנים.

? היכן לדעתך יש למקם את נתוני הגרפיקה של השחקן השני?

ובכן, השחקן השני, שחקן מס' 1, בנוי גם הוא מ-128 בתים, הממוקמים החל מהמקום ה-640 מתחילת ה-PMBASE.

שים לב, הבית השולט על מיקומו האופקי של השחקן השני הוא 53249, והבית השולט על צבעו הוא: 705.

■ נסה לכתוב בעצמך התוכנית שתציג זה לצד זה את השחקן האיש ואת השחקן החייזר. עבוד הפעם במסך גרפי מס' 3 וצבע את שני השחקנים שלך בצבעים שונים (החייזר יהיה למשל בצבע ירוק חייזרי ואילו האיש – בצבע כחול ארצי)

אל תשכח לנקות בתחילת הדרך את האזור הרלוונטי בתוך זיכרון השחקנים.

(תשובה 8, בעמ' 64)

נזיז עכשיו את החייזר כך ש"יטפס" על האיש.

■ פקוד על המחשב:

```
POKE 53249, 100
```

החייזר אמנם זז אבל הוא מתנהג לא בדיוק כפי שרצינו – במקום שהחייזר יטפס על האיש, האיש מטפס על החייזר!

■ עתה פקוד על המחשב:

```
POKE 623, 33
```

עכשיו קיבלנו את התוצאה הרצויה.

שים לב: בית 623 קובע כיצד יתנהגו השחקנים, זה ביחס לזה, וביחס לאלמנטים הנוספים על המסך.

■ החזר את החייזר למקומו ההתחלתי והוסף לתוכנה שלך את השורה הבאה:

```
15 COLOR 1: PLOT 12,10: DR. 32,10
```

כצפוי הופיעה על המסך "גדר" כתומה והשחקנים עומדים לפניה.

■ פקוד עתה על המחשב:

```
POKE 623,4
```

כעת החייזר עבר שכבה אחורה והוא מסתתר מאחורי הגדר.

■ פקוד על המחשב:

```
POKE 623, 8
```

לכאורה אין שום שינוי.

■ אך ההבדל יתגלה לפניך אם תשנה את מספר הצבע בשורה 15 ותקליד:

```
15 COLOR 3: PLOT 12,10: DR. 32,10
```

נסכם את מה שראינו עד כה לגבי התנהגותם של שני השחקנים: 0, 1 כתלות בערכו של בית 623.

הערך 1 ממקם את השחקנים בקדמת הגרפיקה (למשל, מעל הגדר שבנית) – שחקן 0 ממוקם לפני כולם, ומאחוריו השחקן הבא בתור – 1
הערך 33 ממקם את השחקנים בקדמת הגרפיקה – שחקן 1 ממוקם לפני כולם, ומאחוריו שחקן מספר 0
הערך 4 ממקם את שני השחקנים מאחורי הגרפיקה.

הערך 8 ממקם את שני השחקנים בקדמת הגרפיקה, אך רק כאשר הגרפיקה צבועה בצבע מס' 1 או 2. גרפיקה הצבועה בצבע מס' 3 נשארת מקדימה.

כדי להשלים את תמונת השחקנים, נוסיף עתה עוד שני חייזרים למשחק – חייזר מס' 2 בצבע אדום וחייזר מס' 3 בצבע סגול.

אנו ממליצים לך לחשב בכוחות עצמך (באמצעות מפת השחקנים) את מיקומם של שני השחקנים הנוספים, ולהוסיף את שורות הקוד המתאימות מבלי להציץ בתשובות שבסוף החוברת.

שים לב, על מיקומם האופקי של השחקן השלישי והרביעי שולטים, בהתאמה, הבתים 53250 ו-53251, ואילו על צבעיהם שולטים, בהתאמה, הבתים: 706 ו-707.

ועוד טיפ קטן לפני שתיגש למשימה – מכיוון שמדובר בשלושה חייזרים זהים, אין צורך להקליד פעמיים נוספות את נתוני הגרפיקה שלהם – ותוכל להשתמש לצורך כך בפקודה: **RESTORE**.

בהצלחה!

(תשובה 9, בעמ' 65)

■ עתה פקוד על המחשב:

```
POKE 623, 1
```

במצב זה, כצפוי, כל ארבעת השחקנים עומדים לפני החומה.

כיצד יתנהגו השחקנים אם תפקוד:

```
POKE 623, 4
```

כמובן כולם יתייצבו כאיש אחד מאחורי החומה.

ומה לגבי:

```
POKE 623, 8
```

תלוי בצבע הגרפיקה! אם החומה נצבעה בצבע 3, הרי שכל השחקנים יתייצבו מאחוריה, אך אם היא נצבעה בצבע 1 או 2, כל השחקנים יתמקמו מקדימה.

? מה לדעתך יקרה אם תפקוד על המחשב:

```
POKE 623, 2
```

נסה!

במצב זה השחקנים נחלקים לשתי קבוצות: שחקנים 0 ו-1 עומדים מלפנים ואילו שחקנים 2 ו-3 עומדים מאחור.

כשתתחיל לתכנת בעצמך משחקי מחשב תוכל לעשות שימוש משכיל בידע שלמדת. לפעמים תרצה שהשחקנים שלך יעברו לפני אלמנטים מסוימים על המסך, ולפעמים תרצה שיסתתרו מאחורי אלמנטים אחרים. באמצעות הכנסת ערכים מתאימים אל תוך בית 623 תוכל להתמודד בקלות עם צרכים תכנותיים אלה.

עד כאן, לעת עתה, העיסוק בשחקנים (PLAYERS). אך עדיין לא טיפלנו בכל ה-PMG. נותר לנו לדון בטילים (MISSILES). בכך נעסוק בפרק הבא.

פרק ג
משגרים טילים

במחשב האטארי שלך יש כאמור ארבעה "טילים" – מִינִי-שחקנים שיכולים לנוע על המסך בנפרד, או לצד ארבעת השחקנים.

צורת הטיפול בטילים דומה לצורת הטיפול בשחקנים, ועליך לזכור כי ארבעת הטילים הם למעשה "שחקן" אחד בפני עצמו שיש לו מעין "פיצול אישיות". כל "אישיות" שכזו נדלקת בהתאם לערכי המידע הגרפי שמוכנסים אל זיכרון השחקן.

בפרק הראשון למדת כי רוחבו של שחקן הוא 8 ביטים (פיקסלים), וכי שיטת חישוב ערך המידע הגרפי היא "שיטת המשקלים":

128	64	32	16	8	4	2	1

אם נפצל את השורה הזו לארבעה חלקים שווים, נקבל ארבעה מקטעים בני שני ביטים, כל אחד:

128	64		32	16		8	4		2	1
טיל 3			טיל 2			טיל 1			טיל 0	

הטיל הראשון, טיל מס' 0 יכול לקבל את אחד משלושת הערכים: 1, 2, 3

או שרק ה"ביט" הימני דולק, או שרק השמאלי דולק, או ששני ה"ביטים" דולקים (ואז הטיל "רחב").

? מהם הערכים שיכולים לקבל כל אחד משלושת הטילים האחרים?

טיל מס' 1: ____, ____, ____

טיל מס' 2: ____, ____, ____

טיל מס' 3: ____, ____, ____

(תשובה 10, בעמ' 66)

נחזור עתה אל מפת זיכרון השחקנים בעמוד 12.
המידע המוקצה לארבעת הטילים מתחיל בנקודה PMBASE+384 ומסתיים בנקודה PMBASE+512
בתוך טווח זה של 128 בתים, אפשר לאכסן מידע עבור טיל אחד, שניים, שלושה או ארבעה.
אם נציב בתוך התאים הללו מידע מסוג (1/2/3) רק הטיל הראשון "ידלק" על המסך.
אם נציב בתוך התאים הללו מידע מסוג (4/8/12) רק הטיל השני "ידלק" על המסך.
וכך הלאה...

כדי להבין טוב יותר את מבנה הטילים, נפתח ביחד איתך את המשחק הראשון שלך בו נעשה שימוש בגרפיקת השחקנים.

ובכן, מסתבר שהחייזר הירקרק שהגיע אל כדור הארץ הוא חייזר עוין ויש להשמידו לאלתר. לצורך המשימה נבחר השחקן האנושי הגיבור שלנו, ובידיו כלי נשק קטלני.

את החייזר נריץ לרוחב המסך בחלקו העליון במהירות קבועה, ואילו הגיבור שלנו ינוע ימינה ושמאלה בתחתית המסך, באמצעות הג'ויסטיק.
לחיצה על כפתור הג'ויסטיק תשחרר טיל בודד שינוע מעלה וינסה לפגוע בחייזר. אם המשימה תוכתר בהצלחה, החייזר יהבהב בשלל צבעים ואז יושמד.

כרגיל, אנו מציעים לך לנסות להתמודד בעצמך עם כתיבת התוכנה.

■ נסה! **אך בכל מקרה עיין גם בהסבר המפורט שלנו.**

שלב א' – בונים את השחקנים (האיש, החייזר והטיל).

- נכנה את קואורדינטת האיש XM, YM; את קואורדינטות החייזר: XE, YE, ואת קואורדינטות הטיל XT, YT. כדאי למקם את יציאת הטיל הנורה אל על, מעט מעל גובהו של השחקן.
- ננקה את אזור הטילים והשחקנים 0, 1.
- מומלץ להגדיר תמיד את התנהגותם של השחקנים (באמצעות בית 623)

```
10 A=PEEK (106)-4: PMBASE=A*256:
POKE 106, A: GR. 0: POKE 54279, A
20 XM=100: YM=100: XE=50: YE=30: YT=97
30 POKE 53277, 3: POKE 559, 46: POKE 623, 1
40 FOR R=PMBASE+384 TO PMBASE+768:
POKE R, 0: NEXT R
```

```
50 FOR R=PMBASE+512+YM TO PMBASE+512+YM+11:
READ M: POKE R,M: NEXT R
60 FOR R=PMBASE+640+YE TO PMBASE+640+YE+9:
READ M: POKE R,M: NEXT R
70 POKE 704,130: POKE 705,168
1000 DATA 0,56,56,16,56,92,154,24,60,
36,36,0
1010 DATA 0,60,126,219,126,36,36,66,
129,0
```

שלב ב׳ – מוסיפים את תנועת הג׳ויסטיק עבור השחקן ואת תנועתו המחזורית של החייזר.

בסוף פרק א׳ עמדנו על בעיית האיטיות בתנועות השחקן. אחת הדרכים להאיץ מעט את עבודת המחשב היא להעדיף פניות ישירות לכתובות זיכרון, על פני פקודות.

כך למשל במקום להשתמש בפקודה STICK (0) תוכל להשתמש בכתובת הזיכרון: PEEK (632)

ו- PEEK (633) הוא המקבילה לפקודה STICK (1)

מכיוון שבמשחק שלנו הגיבור נע רק על הציר האופקי (ימינה ושמאלה) אפשר להוסיף פקודת תנועה אחת המכילה בתוכה גם הגבלת תנועה:

```
80 XM=XM+((PEEK (632)=7)*(XM<212))-
((PEEK (632)=11)*(XM>35)): POKE 53248,XM
```

נסביר:

תנועת השחקן נבנית משני חלקי נוסחה – החלק השמאלי של הנוסחה מטפל בתזוזת הג'ויסטיק ימינה ולכן הוא מגדיל את קואורדינטת ה-X ביחידה אחת, ואילו החלק הימני של הנוסחה מטפל בתזוזת הג'ויסטיק שמאלה, ולכן הוא מקטין את קואורדינטת ה-X ביחידה אחת. כמובן ששני חלקי הנוסחה לא יכולים להתרחש במקביל!
בנוסף, לכל צד של הנוסחה הוספנו תנאי לוגי הבוחן את גבול הקואורדינטה. שלילת התנאי הלוגי מאפסת את ההקטנה או ההגדלה של הקואורדינטה.

את החייזר נניע בקלות, ובמקרה של החייזר, הגבלת התנועה שלו כשלעצמה יוצרת את המחזוריות המעגלית:

```
90 XE=XE+2: POKE 53249,XE: IF XE>212
THEN XE=35
120 GOTO 80
```

אגב, כפי ששמת לב הכפלנו את קצב ההתקדמות של החייזר פי 2 והתנועה שלו עדיין נראית חלקה. תוכל כמובן לחזור לשורה 80 ולהכפיל גם את קצב ההתקדמות של הגיבור (אל תשכח לעשות זאת בשני חלקי הנוסחה)

שלב ג' – מוסיפים את ירי הטיל

את הטיל כזכור, אנו רוצים לשגר מעלה באמצעות לחיצה על כפתור הג'ויסטיק. שוב, על מנת להאיץ מעט את פעולת המחשב, נעדיף פנייה לכתובת זיכרון על פני השימוש בפקודה: במקום להשתמש בפקודה **STRIG (0)** נוכל להשתמש בפקודה המקבילה: **PEEK (644)**
[ובפקודה **PEEK (645)**, במקום **STRIG (1)**]

כדי למנוע אפשרות לשגר כמה טילים בעת ובעונה אחת, נוסיף משתנה עזר בשם TIL שיקבל ערך 1 כאשר הטיל באוויר, ו-0 כאשר הטיל איננו באוויר. בשעה שהטיל נורה אל על, נקבֵּעַ גם את מקומו האופקי XT בהתאם למיקום **מרכז** דמות השחקן **ברגע** השיגור.

```
100 IF (TIL=0) AND (PEEK(644)=0) THEN
TIL=1: XT=XM+3
110 IF TIL=1 THEN GOSUB 200
```

הגיע הזמן לעצב את הטיל ולשגרו כלפי מעלה.

מכיוון שממדיו של הטיל קטנים, אין הצדקה לעצב אותו בכל פעם מחדש תוך פנייה לקריאת ה-DATA שלו. יעיל ומהיר הרבה יותר להציב את ערכי המידע הגרפי שלו ישירות במיקומם המתאים. אגב, מכיוון שתנועת הטיל היא מלמטה למעלה, אין צורך לנקות (לאפס) בכל פעם את השורה העליונה, אלא רק את התחתונה (או את שתי השורות התחתונות, אם תנועת הטיל מהירה יותר, בקפיצות של שני פיקסלים). כך, לטיל מהיר בעובי שתי שורות, נדרשים ארבעה ערכים בלבד. את מקומם של ארבעת הערכים הללו נעתיק שוב ושוב עד לגובה המרבי של המסך.

```
200 YT=YT-2: IF YT<1 THEN TIL=0: YT=97:
RETURN
210 POKE 53252,XT: R=PMBASE+384+YT: POKE
R,3: POKE R+1,3: POKE R+2,0: POKE R+3,0
```

לבסוף, נותר לנו לבדוק אם התרחשה "פגיעה" במטרה. בפרק הבא נלמד דרך יעילה ופשוטה לבחון התנגשויות בין שחקן לשחקן (ובין שחקן לטיל). בינתיים נורה למחשב לבדוק פגיעה באמצעות השוואה בין הקואורדינטות של הטיל לבין הקואורדינטות של החייזר. מכיוון שהחייזר תופס מרחב של

8X8 עלינו לערוך את ההשוואה ביחס לטווח של קואורדינטות ולא ביחס לקואורדינטה בודדת. לשם כך נוכל להשתמש בפונקציה **ABS** (ערך מוחלט):

```
220 IF ABS (YT-YE)<4 AND ABS(XT-XE)<4
THEN 300
230 RETURN
```

נוסיף את אפקט ההבהוב של החייזר בעת הפגיעה:

```
300 POKE 53252,0: FOR R=1 TO 255:
POKE 705,R: NEXT R: POKE 53249,0
310 ? "CONGRATULATIONS! YOU SAVED EARTH":
END
```

טיל מול טיל

עכשיו כשסיימת לבנות את המשחק הראשון שלך בגרפיקת השחקנים, הגיע הזמן לשפר אותו ולהוסיף לו אלמנט אתגרי. את החייזר העוין שלנו נצייד בנשק קטלני לא פחות – טילים מהירים שנורים כלפי מטה במטרה לפגוע בשחקן הגיבור.

זכור כי ארבעת הטילים "יושבים" על אותו אזור זיכרון ועל מנת "להדליק" את הטיל הרצוי יש להציב בתוך אזור זה את הערכים הרלוונטיים עבורו. לכן, אם עבור הטיל הנורה מעלה הצבנו את הערך 3 (שהוא צירוף של שני ה"ביטים" 1 ו-2), עבור הטיל שיורה החייזר כלפי מטה נציב את הערך 4 או 8 או 12.

אנו ממליצים להכפיל את מהירות הטילים של החייזר פי 3 (זכרו להוסיף בראש הטיל שלושה ערכי 0). עוד אנו ממליצים שהחייזר יירה את הטילים באופן רנדומלי.

■ הוסף עתה בעצמך את שורות הקוד הנדרשות. כרגיל, אנו מציעים לך את הפתרון בפרק התשובות.

(תשובה 11, בעמ' 66)

לעתים תרצה לכתוב משחקים שנורים בהם יותר משני טילים. תוכל "להדליק" כל אחד מארבעת הטילים הקיימים כפי שכבר למדת, וכמובן שתוכל גם להזיזם לרוחב המסך בקלות. אלו הם הבתים השולטים בתנועת הטילים:

53252 – אחראי על טיל 0

53253 – אחראי על טיל 1

53254 – אחראי על טיל 2

53255 – אחראי על טיל 3

■ נסה לשכלל את המשחק ולהוסיף לו עוד שני חייזרים עוינים שיורים אף הם טילים על הגיבור שלנו.

מה דעתך להניע את החייזרים בכיוונים מנוגדים? לצייד אחד מהם בטיל מהיר יותר מן הטילים של חבריו?

תוכל להוסיף גם ניקוד למשחק, צלילים, ועוד כהנה וכהנה. אנו משאירים לך את מלאכת הפיתוח והתכנות!

מצרפים שחקן חמישי

לפני שנעבור לפרק הבא, נדגיש עוד עניין לגבי הטילים.
עד עתה ראית כי הצבת הערכים 1, 2, 3 מדליקה את טיל מס׳ 0; הצבת הערכים 4, 8, 12 מדליקה את טיל מס׳ 1, וכן הלאה.

? מה לדעתך יקרה אם נציב בתוך אזור מידע הטילים את הערך 15?

מכיוון ש-15 הוא הסכום של 12+3, יידלקו בעת ובעונה אחת שני טילים ״מלאים״: טיל 0 + טיל 1

? מה יקרה אם נציב את הערך 80? ומה לגבי הערך 255?

ובכן, הערך 80 ידליק בעת ובעונה אחת שני טילים צרים: טיל 2 וטיל 3
הערך 255 ידליק בעת ובעונה אחת את כל ארבעת הטילים המלאים.

כל זה מוביל אותנו למסקנה הבאה: תוכל לבנות **במקום** ארבעת הטילים, שחקן שלם נוסף, שישמש כ״שחקן חמישי״. על מנת לעצב את ה״שחקן״ הזה השתמש באותה שיטה בה עיצבת את דמותו של כל אחד מארבעת השחקנים (קרי: שיטת ״המשקלים״/סיביות).

זכור רק שכדי למקם את השחקן החמישי על המסך, תצטרך לתאם ולפקוד בו זמנית ערכים לתוך כל אחד מארבעת הבתים השולטים על מקומו האופקי של כל טיל. כך למשל, אם תרצה להציב את השחקן החמישי שלך על קואורדינטה 100, הצב בבית 53252 את הערך 100, בבית 53253 את הערך 98, בבית 53254 את הערך 96, ולבסוף בבית 53255 את הערך 94.

ולבסוף, ודאי שמת לב כי אין אפשרות לשלוט בצבעו של כל טיל. ארבעת הטילים מקבלים, כל אחד, את צבעו של השחקן התואם לו, כלומר טיל 0 יקבל את צבעו של שחקן 0, טיל 1 יקבל את צבעו של שחקן 1, וכן הלאה.

יחד עם זאת, מכיוון שהמחשב מאפשר להתייחס אל ארבעת הטילים כאל שחקן נוסף בפני עצמו, ניתנת גם האפשרות לצבוע את כל ארבעת הטילים בצבע אחיד אחד.

על מנת לעשות זאת, בא לעזרתנו בית 623 (אשר כבר דנו בו בפרק הקודם). אם תציב בתוכו את הערך 17, יקבלו כל ארבעת הטילים (קרי: "השחקן החמישי") צבע חדש אחד.
זאת ועוד זאת, את הצבע הזה תוכל לשנות כרצונך באמצעות הבית 711.

■ נסה לבנות בעצמך שחקן (למשל את הגיבור או החייזר) על ידי שימוש בארבעת הטילים. צבע את השחקן החמישי שלך בצבע אחיד ונסה להניע (באמצעות הג'ויסטיק) לרוחב המסך.

אם תתקשה במשימה, תוכל לגשת לפרק האחרון, ובו הדגמה משעשעת של אפשרויות השימוש בכל הטילים והשחקנים גם יחד.

בפרק הבא נדבר על התנגשויות. אל תשכח לחגור את חגורת הבטיחות...

פרק ד
התנגשויות

בתוך מעבד ה-ANTIC של האטארי – השולט בגרפיקת השחקנים – מוקצים כמה בתים ייעודיים לטיפול בכל נושא ההתנגשויות בין שחקנים לבין טילים, ובין שחקנים לבין עצמם.
השימוש בבתים אלו פשוט ביותר. הוא חוסך תכנות מסורבל של השוואת טווחי קואורדינטות (כפי שעשית בפרק הקודם) והכי חשוב, הוא מקצר משמעותית את זמן פעולת המחשב – אשר כפי שראית, חיוני ביותר ליצירת תנועה חלקה ומהירה יותר של השחקנים.

ראשית, נקדים ונספר לך כי טווח הבתים מ-53248 ועד 53263 מיועד לבחינת כל סוגי ההתנגשויות האפשריות. האם הבתים הללו נראים לך מוכרים? בוודאי! למעשה את בית 53248 פגשת כבר בתחילת הפרק הראשון, וכזכור לך, הוא מאפשר לשנות את מיקומו האופקי של השחקן הראשון.

כרגיל, לא ניכס לעובי הקורה של מה מתחולל בתוך קרבי המחשב שלך, אלא נפשט ונאמר כי הבית 53248, כמו גם יתר הבתים בטווח הנ״ל, ״מועסקים בשתי עבודות״ – גם כ״טריגרים״ הפוקדים על המחשב לבצע פעולות, וגם כבתים ״מנטרים״ שאפשר לקרוא מתוכם ערכים ולהסיק באמצעותם מסקנות. על מנת להבין כיצד משתמשים בבתים הללו בתפקידם השני, כ״בתים מנטרים״ – נחזור אל החייזר ואל הגיבור שלנו.

לנוחיותך נציג שוב את התוכנה המציירת על המסך את שני השחקנים – הגיבור והחייזר. הפעם נמקם את שניהם באותו גובה, זה מול זה, וכך כשנריץ את התוכנה, החייזר ינוע מן הקצה השמאלי של המסך אל עבר הקצה הימני וחוזר חלילה, כשבדרך יעבור **מאחורי** הגיבור (בהתאם לערך שמקבל תא 623)

```
10 A=PEEK (106)-4: PMBASE=A*256:
POKE 106,A: GRAPHICS 0: POKE 54279,A
20 XM=100: YM=70: XE=50: YE=70
30 POKE 53277,3: POKE 559,46: POKE 623,1
40 FOR R=PMBASE+384 TO PMBASE+768:
POKE R,0: NEXT R
50 FOR R=PMBASE+512+YM TO PMBASE+512+YM+11:
READ M: POKE R,M: NEXT R
60 FOR R=PMBASE+640+YE TO PMBASE+640+YE+9:
READ M: POKE R,M: NEXT R
70 POKE 704,130: POKE 705,168
80 XM=XM+2*((PEEK(632)=7)*(XM<212))-
2*((PEEK(632)=11)*(XM>35)): POKE 53248,XM
90 XE=XE+2: POKE 53249,XE: IF XE>212
THEN XE=35
130 GOTO 80
1000 DATA 0,56,56,16,56,92,154,24,60,
36,36,0
1010 DATA 0,60,126,219,126,36,36,66,
129,0
```

עד כאן, הכול מוכר לך.

■ הוסף עתה את שתי השורות הבאות:

```
100 P=PEEK (53261): SOUND 1,20,2,6*P
110 POKE 53278,1
```

הרץ את התוכנית וראה מה קורה ברגע שהחייזר נפגש עם הגיבור.

שים לב: בית 53261 בודק אם שחקן מספר 1 (החייזר, במקרה שלנו) התנגש עם שחקן אחר. כל עוד לא התרחשה התנגשות, נשמר בבית הזה הערך 0. ברגע שמתרחשת התנגשות משתנה ערכו של הבית ל-1 וכך הוא נשאר (גם כאשר השחקנים ממשיכים לנוע ונפרדים זה מזה!).

באותה מידה, אגב, יכולנו לכתוב גם את השורות הבאות:

```
100 P=PEEK (53260): SOUND 1,20,2,6*P
110 POKE 53278,1
```

שים לב: בית 53260 בודק אם שחקן 0 (הגיבור, במקרה שלנו) התנגש עם שחקן אחר. מכיוון שההתנגשות היא כמובן הדדית, מאותו הרגע שבו התרחשה ההתנגשות, גם ערכו של הבית הזה הופך מ-0 ל-1.

בשורה מספר 110 אנו מכניסים לפעולה בית ייחודי שמספרו 53278 – בית זה מאפס מחדש את ערכם של הבתים 53260 ו-53261, כלומר בלשון ציורית, גורם לחייזר ולגיבור "לשכוח" שאי פעם התרחשה ביניהם התנגשות.

שים לב: כדי לאפס את ערכי הבתים 53260 ו-53261, יש לפקוד אל תוך בית 53278 את הערך 1 (ולא 0)!

ומה באשר לשני השחקנים הנוספים? ובכן, בית 53262 מנטר את פעילותו של שחקן מספר 2 ואילו בית 53263 מנטר את פעילותו שחקן מספר 3.

גם כאן, יישמר בבתים הללו הערך 0 כל עוד לא התרחשה התנגשות, ומרגע שמתרחשת התנגשות ישתנה ערכו של הבית הרלוונטי ל-1.

כיצד מאפסים את הבתים 53262 ו-53263? – כמובן באמצעות הבית הייחודי 53278, כאשר נפקוד לתוכו את הערך 1.

התנגשויות נוספות

■ מחק את שורות מספר 90, 100, 110 מן התוכנית; שנה (בשורה מספר 10) את המסך למסך הגרפי מספר 3 והוסף את השורה:

```
15 COLOR 1: PLOT 30,10: DR. 30,20
```

הרץ את התוכנית. כצפוי הופיעה חומה גבוהה בצבע כתום במרכז המסך, ואם תזיז באמצעות הג׳ויסטיק את השחקן לכיוונה, הוא יעבור (מעל / מתחת) לחומה.

כשתכתוב משחקי פעולה תרצה שהשחקנים שלך יגיבו לחומות (או לאלמנטים גרפיים אחרים על המסך) ולא תמיד יעברו דרכם.

לשם כך ישנה עוד קבוצה של בתים – 53252, 53253, 53254, 53255. כל אחד מהם מנטר (בהתאמה) את ארבעת השחקנים ובודק התנגשויות בינם לבין אלמנטים על המסך.

■ פקוד למשל את השורה הבאה:

```
90 P=PEEK (53252): IF P<>0 THEN XM=XM-2:
POKE 53278,1
```

שים לב: בית 53252 בודק את ההתנגשות בין שחקן מספר 0 (הגיבור שלנו) לבין הגרפיקה שעל המסך (ובמקרה שלנו, הקיר הכתום). מרגע שהתרחשה התנגשות משתנה ערכו של הבית מ-0 לערך אחר, וכזה הוא נשאר עד אשר פוקדים לתוך הבית 53278 את הערך 1.

דע לך כי הערך שנשמר בתוך הבית 53252 בשעה שמתרחשת התנגשות, מלמד לא רק על כך שאירעה התנגשות אלא גם על טיב ההתנגשות. הערך 1 מלמד

על כך שהגיבור התנגש בקיר שצבעו 1, ערך 2 מלמד על כך שהגיבור התנגש בקיר שצבעו 2 וכן הלאה.

תוכל להשתמש במידע זה כשתבנה משחקים מורכבים יותר. למשל, אם תרצה שחלק מהקירות יהיו אלסטיים (כאלו שהשחקן יכול לגעת בהם אך לא לעבור אותם) ואחרים יהיו קטלניים (כאלו שאם השחקן נוגע בהם הוא נפסל).

התנגשויות בין טילים לבין שחקנים

כדי לבנות משחקי מלחמה, כמובן שנצטרך דרך יעילה לבדוק פגיעה (התנגשות) של טילים בשחקנים. לצורך כך משמשים אותנו הבתים 53256, 53257, 53258, 53259. כל אחד מהם מנטר את ההתנגשות בין כל אחד מארבעת הטילים (בהתאמה) עם השחקנים.

ניקח למשל את בית מספר 53256. בית זה בודק האם טיל מספר 0 התנגש עם אחד השחקנים. כל עוד לא התרחשה התנגשות, נשמר בבית הזה הערך 0. ברגע שמתרחשת התנגשות משתנה ערכו של הבית ל-2 וכך הוא נשאר (גם כאשר הטיל ממשיך הלאה ו"נפרד" מן השחקן).

■ הוסף עתה את השורות הבאות:

```
100 XT=XT+1: POKE PMBASE+384+YM+5,12:
POKE 53253,XT
110 IF XT>210 TEHN XT=0
```

כצפוי, נורה שוב ושוב טיל מצדו השמאלי של המסך לעבר צדו הימני (טיל מספר 1).

■ שנה כעת את שורה מספר 90 כדי שתבחן אם הטיל פגע בגיבור ותגרום לו לדמם (להיצבע בצבע אדום), אם אכן הגיבור נפגע...

(תשובה 12, עמ' 67)

? ומה אם נרצה לאפשר לגיבור שלנו לברוח מן הטיל ולהסתתר מאחורי הקיר?

כאן נכניס לפעולה את אחד מהבתים: 53248, 53249, 53250, 53251. כל אחד מהם מנטר את ההתנגשות בין כל אחד מארבעת הטילים (בהתאמה) עם אלמנטים גרפיים על המסך.

■ הוסף עתה שורה נוספת (שורה מספר 120) לתוכנית כדי שתציל את הגיבור שלנו ותגרום לטיל "להיספג" בקיר (כלומר להיעלם). אל תשכח לאפס את ההתנגשות בין הטיל לבין הקיר, בטרם ישוגר הטיל הבא...

(תשובה 13, עמ' 67)

עד כאן בנושא ההתנגשויות.

נדגיש שוב את היתרון העצום של השימוש בבתים המנטרים לבחינת התנגשויות. שים לב כי בתים אלו רגישים אך ורק לפיקסלים הדולקים של השחקנים. נתבונן למשל על הגיבור שלנו – אם נורים לעברו שני טילים – האחד בגובה מרכז הגוף, והשני בגובה הראש, הטיל המכוון למרכז הגוף יפגע בו ראשון, וזאת מכיוון שמרכז גופו של הגיבור רחב יותר מראשו. כך מתאפשר לנו בדרך אלגנטית וחסכונית מאוד לבנות משחקים בעלי רמת דיוק גבוהה.

מוכן להגביר את הקצב? שלב להילוך חמישי ועבור לפרק הבא!

פרק ה
מגבירים את הקצב

בפרקים הקודמים למדת לנצל את מלוא יכולותיו הגרפיות של מעבד ה-ANTIC במחשב האטארי שלך. דע לך כי ארבעת השחקנים, כמו גם ארבעת הטילים יכולים לשמש אותך לצרכים רבים ומגוונים, ולא רק לבניית משחקי פעולה.

באמצעות נתוניהם הגרפיים של השחקנים תוכל ליצור איורים וצורות ולשלב אותם עם אלמנטים אחרים במסך הגרפי שבו אתה עובד. במקום "גיבור" או "חייזר" תוכל ליצור למשל לוגו או חתימה שלך ולמקם אותם בראש המסך או בתחתיתו.

- נסה!

במסכים הגרפיים בהם המחשב מגביל אותך בכמות הצבעים, תוכל לעקוף את המגבלה, וליצור באמצעות ה-PMG צורות צבעוניות ולשלבן בתמונה הכוללת. זכור כי יכולים לעמוד לרשותך עוד 5 צבעים (ארבעה צבעי השחקנים, וצבע נוסף, אם תשתמש בארבעת הטילים כשחקן חמישי)

- נסה למשל לצייר במסך הגרפי מספר 3 את כל צבעי הקשת – שישה קווים אנכיים בצבעים: סגול, כחול, ירוק, צהוב, כתום ואדום!

האפשרויות הגרפיות הפתוחות עתה לפניך רבות מאוד. עבור כל הצרכים הגרפיים הללו, אנו ממליצים לך להשתמש תמיד בטכניקה שלמדת: להוריד את ה-RAMTOP ולהפנות את מעבד ה-ANTIC שלך לאזור הזה בזיכרון.

אבל אם תרצה לפתח בשפת הבייסיק משחקי פעולה משוכללים יותר, שיש בהם תנועה לא רק על הציר האופקי, אלא גם על הציר האנכי, כדאי שתאמץ גישה חדשה לשימוש ב-PMG. בסוף פרק א' הבטחנו להציע לך דרכים להאיץ את תנועתם של השחקנים, והגיע הזמן שנעמוד בהבטחתנו!

ראשית, נזכיר שוב את הבעיה:

בניגוד לתנועה ימינה ושמאלה של השחקן, שאחראי לה בית אחד בלבד (בית 53248 למשל אחראי על שחקן 0), התנועה למעלה ולמטה מחייבת העתקה שוב ושוב של כל נתוני השחקן בזה אחר זה למיקומם החדש (כולל הצורך לאפס שורה אחת או יותר בקצה המנוגד לכיוון התנועה).

כדי להתמודד עם הבעיה, עלינו למצוא טכניקה מהירה ביותר להעתקת נתונים ממקום למקום, כמו גם טכניקה לאיפוס טווחים גדולים של זיכרון. למזלנו הטכניקה הזו מסתתרת באחד מן הכלים הבסיסיים ביותר בשפת הבייסיק – המשתנים המחרוזתיים.

אם תגדיר למשל שני משתנים מחרוזתיים: **A$** ו-**B$** בגודל של 128 בתים כל אחד, ותציב מחרוזת ארוכה בתוך המשתנה הראשון, ההעתקה של המידע מהמשתנה האחד למשתנה השני תתבצע כהרף עין.

■ נסה!

כזכור, כדי לאכסן את כל המידע עבור גרפיקת השחקנים דרושים לנו 1024 בתים. נוכל להגדיר בנקל שישה משתנים מחרוזתיים בגדלים מסוימים (בהתאם למפת השחקנים), ולאכסן בתוכם את כל המידע הדרוש ל-PMG.

אנו ממליצים לך לכנות את המשתנים הללו בשמות קצרים וברורים:

```
50 DIM BUFFER$(384), MSLS$(128), PL0$(128),
PL1$(128), PL2$(128), PL3$(128)
```

ה-BUFFER הוא אותו חלק "לא בשימוש" בן 384 בתים שיש להשאיר ריק, אחריו מקצים 128 בתים עבור כל הטילים, ובהמשך 128 בתים לכל אחד מארבעת השחקנים – ובסך הכול 1024 בתים.

לתוך כל אחד מהמשתנים המחרוזתיים שהגדרנו אפשר להכניס בנקל את נתוני הגרפיקה של השחקנים.

כך למשל, את נתוני הגיבור: 0, 36, 36, 60, 24, 154, 92, 56, 16, 56, 56, 0 אפשר להכניס אל תוך המשתנה **PL1$**, ונעשה זאת כמובן באמצעות המרת הנתון לערך ה-ASCII שלו: ♥$$<┴▙\8♣88♥

עד כאן הכול טוב ויפה. אולם איך נוכל ללמד את המחשב לקרוא את נתוני גרפיקת השחקנים שלנו מתוך המשתנים המחרוזתיים הללו?

כאן באה לעזרתנו הפקודה **ADR**.

בפרק הראשון של מחשבת 6 למדת כי הבית הראשון המוקצה לשמירת תוכניות הבייסיק הוא 2049, וכי קצה טווח הזיכרון האופציונלי (כל עוד לא הורדת את ה-RAMTOP) הוא 40960 (במחשב אטארי 800XL). בתוך טווח הזיכרון הזה (כ-38 קילו-בייט) נשמרים בזיכרון המחשב כל שורות הקוד של תוכנית הבייסיק, כמו גם כל המשתנים המספריים, המחרוזתיים והמערכיים שעשית בהם שימוש.

חשוב שתזכור כי הזיכרון הזה הוא **זיכרון דינאמי**. כל שורת קוד שתוסיף (או תמחק) וכל משתנה חדש שתוסיף ישנו את המידע השמור בבתים אלו

של הזיכרון. הפקודה **ADR** יכולה לספר לנו היכן בדיוק – **בכל רגע נתון** – שמור בזיכרון ה-RAM המידע עבור כל משתנה מחרוזתי שהגדרנו.

■ נקה את הזיכרון באמצעות הפקודה **NEW** (חשוב!) והקלד את השורות הבאות:

```
10 DIM A$(10), B$(8), C$(2)
20 ? ADR(A$)
30 ? ADR(B$)
```

התוצאה שקיבלת היא ככל הנראה:

7915

7925

המספר הראשון 7915 הוא תא הזיכרון שבו שמור המשתנה המחרוזתי **A$** והמחשב הקצה עבורו בסך הכול 10 תאים (כמספר המקומות שהגדרנו עבור משתנה זה בפקודת ה- **DIM**).

המספר השני 7925 הוא תא הזיכרון שבו שמור המשתנה המחרוזתי **B$**. מכיוון שעבור משתנה זה הגדרנו רק 8 מקומות, יקצה המחשב עבורו בסך הכול 8 תאים.

■ הוסף עתה את השורה הבאה:

```
40 ? ADR(C$)
```

? לפני שתריץ את התוכנה, חשוב מה לדעתך התוצאה שתתקבל?

התשובה נראית פשוטה למדיי – נראה שמתחת לשני המספרים הנ"ל אמור להופיע מספר שלישי – 7933.

■ נסה!

הפתעה! המחשב משיב מספרים אחרים לגמרי:

7925

7935

7943

מה קרה כאן? ובכן הסברנו כי זיכרון ה-RAM השמור לתוכניות הבייסיק הוא זיכרון דינאמי. ולכן כשהקלדת שורת קוד נוספת, המחשב "טרף את הקלפים מחדש" ותמונת הזיכרון השתנתה כולה.

אולם שים לב – הארגון והסידור של שלושת המשתנים המחרוזתיים לא השתנו כלל ועיקר:
עדיין **A$** הוא הראשון בתור ומוקצים לו בדיוק 10 תאי זיכרון. צמוד אליו עומד המשתנה **B$** ומוקצים לו 8 תאים, והאחרון בתור הוא המשתנה **C$** אשר מוקצים לו 2 תאי זיכרון.

את הידע הזה על אודות מיקומם המדויק של המשתנים המחרוזתיים בזיכרון בכל רגע נתון, נוכל לנצל כדי להגדיר למעבד ה-ANTIC מהיכן לשאוב מתוך משתנים אלו את נתוני השחקנים. במילים אחרות, אחרי שנגדיר כהלכה את המשתנים המחרוזתיים המתאימים ונציב בתוכם את המידע הגרפי הדרוש לנו, נוכל לספק למעבד ה-ANTIC את כתובת הזיכרון המדויקת בו יושבים הנתונים הללו.

ואולם, לפני שנוכל לעשות זאת, עלינו להביא בחשבון מגבלה אחת – מעבד ה-ANTIC אדיש אמנם לגבי המיקום שאנו מקצים עבורו אי שם בטווח תאי ה-RAM, אבל המיקום הזה חייב להיות תָחום בתוך מרווח קבוע של 1024 בתים (כלומר של קילובייט **שלם** אחד).

כך למשל, כתובת הזיכרון 38912 (במחשב אטארי 800XL) – הכתובת שאליה הפנינו את ה-ANTIC עם הורדת ה-RAMTOP – היא כתובת תקינה, כי היא ממוקמת בדיוק קילובייט אחד מתחת ל-RAMTOP. ולכן כשהצבנו בכתובת זו את נתוני השחקנים, פקדנו על המחשב:

```
POKE 106,152
```

? ומה לגבי הפניית ה-ANTIC אל כתובת הזיכרון 38656? (באמצעות הפקודה: **POKE 106,151**)?

ובכן, זו **אינה** כתובת תקינה ומעבד ה-ANTIC לא יבצע את תפקידו כהלכה.

נסכם ונחדד את מה שלמדנו עד כאן:

1. את כל 1024 הנתונים הדרושים למעבד ה-ANTIC אפשר להציב בתוך שישה משתנים מחרוזתיים מוגדרים, כל אחד לפי הגודל הנדרש לו (לפי מפת השחקנים).
2. חייבים להגדיר את המשתנים המחרוזתיים הללו לפי הסדר הנכון (שכן לפי אותו הסדר הם ימוקמו בתוך אזור הזיכרון).
3. מיקומו של המשתנה המחרוזתי הראשון (**BUFFER$**) חייב לשבת על כתובת זיכרון שהיא תחילתו של קילובייט חדש (או במילים אחרות, כתובת הזיכרון של בסיס ה-PMG חייבת להיות כפולה שלמה של 1024).

על מנת לעמוד בתנאי הזה של סעיף 3, עלינו להשתמש בשני משתנים מחרוזתיים נוספים:
משתנה אחד – משתנה "איתור" – בעל גודל מינימלי (של תא אחד) שנגדיר אך ורק כדי לאתר במדויק את כתובתו בזיכרון (באמצעות הפקודה **ADR**).

ומשתנה נוסף – משתנה ״מילוי״ (FILLER), אשר ימלא את החלל בין כתובת הזיכרון שאיתרנו ובין הכפולה הקרובה של 1024.

נדגים:

נניח שכתובת הזיכרון של המשתנה **F1$** (שגודלו בית אחד) היא 8776 ואנו רוצים שהמשתנה **BUFFER$** ישב על הכתובת 9216 (הכפולה הקרובה ביותר של 1024), עלינו להגדיר משתנה עזר נוסף **F2$** בגודל של 339 תאים.
וזאת משום ש: 9216-8776-1=339
(שים לב, הפחתנו בית אחד מגודלו של משתנה המילוי, כדי שהמשתנה העוקב אחריו יתחיל בדיוק בנקודה הרצויה לנו)

■ נסה עתה לכתוב בעצמך את תחילת התוכנה שתכין את זיכרון ה-PMG על גבי משתנים מחרוזתיים ותורה למעבד ה-ANTIC לשאוב נתונים מתוכם.
מצא דרך אלגנטית לחשב את גודלו של משתנה ה״מילוי״ שיביא אותך יחידה אחת מתחת לכפולה הבאה של 1024.

(תשובה 14, עמ׳ 67)

לאחר שהכנו את זיכרון ה-PMG, הגיע הזמן לטעון לתוכו את המידע הגרפי של השחקנים. ראשית, עלינו לנקות (לאפס) את הטווח כולו.

במחשבת 3 למדת, בפרק על מערכים, כי הגדרת משנתה מחרוזתי באמצעות הפקודה **DIM** אינה ״מנקה״ את הזיכרון המוקצה עבור המערך וכי לעתים עשויים להופיע בתוך המערך ״שאריות״ לא רצויות. מסיבה זו המלצנו לך אז לאפס את כל איברי המערך באמצעות לולאת **FOR – NEXT**.

תוכל, כמובן, להשתמש בשיטה זו גם כדי ״לנקות״ את משתני המחרוזת **BUFFER$**, **MSLS$** וכו׳ אך אנו מציעים לך שיטה מהירה יותר, המנצלת את הלוגיקה הפנימית של המחשב.

השיטה פשוטה ביותר: עליך לאפס את המקום הראשון במחרוזת ואת המקום האחרון במחרוזת, ואחר כך להציב מחדש **במקום השני** את המחרוזת כולה.

■ כדי להדגים (וגם להוכיח לך) את השיטה, נמלא את כל איבריה של מחרוזת בת 200 איברים באות A:

```
1 DIM A$(200)
2 A$(1)="A": A$(200)="A": A$(2)=A$
3 PRINT A$
4 END
```

■ לאחר שהתרשמת, ככל הנראה, מן השיטה המהירה, מחק את ארבעת השורות הללו וכתוב בעצמך המשך לתוכנה שלמעלה – שבו מתאפסים כל המשתנים המחרוזתיים.

(המלצה: אחרי שאיפסת את כל ערכיו של משתנה אחד, תוכל לשכפל את ערכיו הריקים לכל שאר המשתנים).

(תשובה 15, עמ' 67)

בשלב הבא נכניס את נתוניהם הגרפיים של השחקנים. לצורך כך, אפשר להגדיר עוד משתנים מחרזותיים – למשל, **GIBOR$** עבור הגיבור ו-**ET$** עבור החייזר.

שים לב – את המשתנים הנוספים הללו, עליך להגדיר אחרי (או לפני) המשתנים המחרוזתיים הקיימים, והעיקר: אין לפגוע ברצף הקיים:

```
F1$, F2$, BUFFER$, MSLS$, PL0$, PL1$, PL2$, PL3$
```

■ נסה להמשיך בעצמך ולכתוב את שורות התוכנה שקוראות את המידע עבור הגיבור והחייזר לתוך תאים מחרוזתיים ייעודיים.

(תשובה 16, עמ' 68)

לבסוף, נציב את השחקן ואת החייזר בתוך מיקומיהם הייעודיים.

■ עשה זאת בעצמך!

(תשובה 17, עמ' 68)

לעת עתה נניח לחייזר ונטפל בתנועת הגיבור. בפרק ב' כבר למדת להניע את הגיבור ימינה ושמאלה באמצעות הג'ויסטיק בצורה אלגנטית ומהירה.
עתה נותר לנו להוסיף לשחקן יכולת לנוע למעלה ולמטה, בתנועה חלקה ומהירה.

■ נסה לכתוב את השורות המתאימות בעצמך. אל תשכח להוסיף הגבלת תנועה גם על הציר האנכי (כדי שהגיבור לא יברח לך פתאום ממסך הטלביזיה אל התקרה...)

(תשובה 18, עמ' 68)

אנו בטוחים שבשלב זה אתה מתרשם ממהירות תנועתו של השחקן על המסך. בידיך עתה כל הידע הנחוץ לך כדי לבנות משחקי פעולה משוכללים ומאתגרים. השתמש בידע שלמדת כדי להניע את השחקנים שלך על פני המסך כולו, לשגר טילים, וליצור אפקטים מרהיבים בעת התנגשויות ביניהם.

■ לפני שנתקדם הלאה, אל הפרק האחרון, עצור לרגע את התוכנית שכתבת ופקוד על המחשב:

```
TEMP=100
```

שים לב למה שהתרחש על המסך. השחקנים שלך קיפצו מעט למטה, וזאת משום שתמונת הזיכרון השתנתה מעט, עם כניסתו של עוד משתנה לזיכרון.

? אגב, מה לדעתך יקרה אם תפקוד על המחשב **NEW**?

התשובה המתבקשת היא שכל השחקנים יעלמו מן המסך באחת, שהרי עם מחיקת התוכנה מן הזיכרון נמחקים גם כל המשתנים המחרוזתיים שהגדרנו. ואולם זכור כי כשם שהגדרת משתנים מחרוזתיים לא מבטיחה שתכלותם תהייה ריקה, כך גם מחיקתם, לא מבטיחה שהמידע שהיה קיים בהם יימחק מן הזיכרון.

■ נסה והיווכח בעצמך!

נסכם ונאמר כי הטכניקה שלמדת בפרק זה, יעילה אך ורק עבור תוכנות בייסיק, כל עוד הן רצות ברקע – ובייחוד אם מדובר במשחקי פעולה. אך אם תרצה להבטיח שהשחקנים שעיצבת יישארו על המסך גם כשאתה עוצר את התוכנית, או טוען לזיכרון תוכנית חדשה, אנו ממליצים לך לחזור לשיטה שלמדת בפרקים הקודמים.

בפרק הבא נסקור בקצרה את אפשרות ההפרדה האופקית של שורה אחת, נלמד אותך עוד אפקט שימושי שתוכל להפעיל על השחקנים שלך, וגם נפתיע אותך בביקור של שלוש דמויות מוכרות, אשר יציגו לפניך דרכים יצירתית לשימוש בגרפיקת השחקנים.

פרק ו
שכלולים נוספים

הפרדה אופקית של שורה אחת

הפרדה אופקית של שורה אחת צורכת כמות כפולה של זיכרון (2048 בתים, שהם 2K, במקום 1024 בתים שהם 1K בלבד). ככלל, **איננו** ממליצים לך להשתמש ברזולוציה גבוהה במשחקי המחשב שתפתח. תוכל ליצור משחקים מלהיבים ומאתגרים גם בהפרדה של שתי שורות, אשר השתמשת בה עד כה.

יחד עם זאת, כבר סקרנו לפניך אפשרויות נוספות לשימוש ב"גרפיקת השחקנים" – כגון איורים ולוגואים. ייתכן שתרצה ליצור איור ברמת חדות גבוהה במיוחד ולשם כך תוכל להשתמש בהפרדה של שורה אחת.

כדי לעשות זאת, עליך לבנות מערך זיכרון כפול בגודלו (כאמור, בן 2048 תאים) ולהכניס לתוכו נתונים עבור הטילים והשחקנים לפי הצד הימני של מפת השחקנים שבעמוד 12. כך למשל, נתוני השחקן הראשון (שחקן מספר 0) יתחילו במקום ה-1024, ויסתיימו במקום ה-1280.

אם אתה שומר את מערך הזיכרון הזה בקצה זיכרון ה-RAM, עליך להוריד את ה-RAMTOP ב-2048 בתים ולא ב-1024. ואם אתה משתמש בשיטה השנייה שלמדת בפרק ה', עליך ליצור משתנה "מילוי" עד לכפולה הבאה של 2048.

ולבסוף, עליך להציב בתוך בית 559 את הערך 62 – וכך להורות למחשב שאתה משתמש בגרפיקה בהפרדה של שורה אחת.

בהמשך הפרק נציג דוגמה לשימוש בהפרדה אופקית של שורה אחת.

פריסה מתוחה ומתוחה כפליים

■ הרץ את אחת התוכנות (מהפרקים הקודמים) המציגה על המסך את החייזר והגיבור, ועצור אותה, עם הופעת שני השחקנים.
פקוד על המחשב:

```
POKE 53256,3
```

עתה פקוד על המחשב:

```
POKE 53256,1
```

ולבסוף החזר את המצב לקדמותו ופקוד:

```
POKE 53256,0
```

שים לב: בתים 53256, 53257, 53258, 53259 – משנים את צורת ההצגה של כל אחד מן השחקנים. הערך 1 מותח את רוחב השחקן, הערך 3 מותח את רוחבו כפליים, ואילו הערך 0 משיב את השחקן לממדיו ההתחלתיים.

גם את רוחבם של הטילים ניתן כמובן למתוח, ולשם כך משמש הבית 53260:

ערכים 1,3 משפיעים על רוחבו של טיל מס׳ 0
ערכים 4,12 משפיעים על רוחבו של טיל מס׳ 1
ערכים 16,48 משפיעים על רוחבו של טיל מס׳ 2
ערכים 64,192 משפיעים על רוחבו של טיל מס׳ 3

? ואיך משנים את רוחבם של שניים, שלושה או ארבעה טילים יחדיו?

מחברים – לפי "שיטת המשקולות" – את הערכים הרצויים.
כך לדוגמה, כדי למתוח במקצת את טיל 0, ובה בעת למתוח כפליים את טיל מס׳ 3, יש לפקוד לתוך בית 53260 את הערך: 1+192=193
כדי למתוח כפליים את כל הטילים, יש לפקוד את הערך: 3+12+48+192=255

דמויות צבעוניות באות לביקור

בפרקים הקודמים למדת כי כל ארבעת השחקנים, ולמעשה גם "השחקן החמישי" הם מונוכרומטיים. כלומר, אפשר לצבוע אותם בכל גוון שתחפוץ, אבל אך ורק כמכלול שלם.

ואולם, אם מצאת לך – בין יחידות הלימוד של התכנות בבייסיק – רגעי פנאי לשחק בכמה ממאות משחקי הפעולה שמציעה חברת "אטארי", ודאי נתקלת בדמויות צבעוניות שמתרוצצות על המסך, ולא רק מונוכרומטיות.

אין לנו כמובן שום יומרה להתחרות במפתחי המשחקים הללו, ואין ספק ששפת התכנות המשמשת אותם פותחת אינספור אפשרויות שאינן קיימות בבייסיק. יחד עם זאת, בעמודים הבאים נציג לפניך טכניקה אחת ליצור דמויות צבעוניות באמצעות שימוש יצירתי בגרפיקת השחקנים.

הרעיון הוא פשוט ביותר. במקום להשתמש בשחקן אחד כדי ליצור את הדמות הרצויה, אפשר להשתמש בשני שחקנים – כל שחקן בונה חלק אחר מן הדמות ונצבע בצבע אחר, וכשהם "מולבשים" זה על גבי זה, הם מתלכדים לדמות צבעונית אחת.

בדוגמה הבאה, השתמשנו בהפרדה של שורה אחת; פתחתנו לשימוש את כל השחקנים והטילים שעומדים לרשותנו, ויצרנו באמצעותם שלוש דמויות:

הדמות הראשונה עושה שימוש בשחקנים 0 ו-1
הדמות השנייה עושה שימוש בשחקנים 2 ו-3
ואילו הדמות השלישית עושה שימוש בארבעת הטילים!

■ הקלד את התוכנה הבאה והשתעשע בתוצאה המפתיעה!

```
10 DIM F1$(1): A1=ADR(F1$):
A2=INT(A1/2048)+1: A3=(A2*2048-A1)-1:
DIM F2$(A3)
20 DIM BUFFER$(768),MSLS$(256), PL0$(256),
PL1$(256), PL2$(256), PL3$(256)
30 DIM BRUC1$(30), BRUC2$(30), SUMO1$(30),
SUMO2$(30), SAMUR$(30)
40 BUFFER$(1)=CHR$(0):BUFFER$(768)=CHR$(0):
BUFFER$(2)=BUFFER$
50 MSLS$=BUFFER$: PL0$=BUFFER$:
PL1$=BUFFER$: PL2$=BUFFER$: PL3$=BUFFER$
60 X1=180:Y1=150:X2=60:Y2=150:X3=90:Y3=153
70 GRAPHICS 0: POKE 53277,3: POKE 559,62:
POKE 623,17
80 PMBASE=ADR(BUFFER$):POKE 54279,
(PMBASE/256)
90 POKE 704,0: POKE 705,25: POKE 706,186:
POKE 707,0: POKE 711,0
100 FOR R=1 TO 23: READ M: BRUC1$(R,R)=CHR$(M):
NEXT R
110 FOR R=1 TO 20: READ M: BRUC2$(R,R)=CHR$(M):
NEXT R
120 FOR R=1 TO 23: READ M: SUMO1$(R,R)=CHR$(M):
NEXT R
```

```
130 FOR R=1 TO 20: READ M: SUMO2$(R,R)=CHR$(M):
NEXT R
140 FOR R=1 TO 20: READ M: SAMUR$(R,R)=CHR$(M):
NEXT R
150 PL0$(Y1,Y1+LEN(BRUC1$))=BRUC1$:
PL1$(Y1,Y1+LEN(BRUC2$))=BRUC2$
160 PL2$(Y2,Y2+LEN(SUMO1$))=SUMO1$:
PL3$(Y2,Y2+LEN(SUMO2$))=SUMO2$
170 MSLS$(Y3,Y3+LEN(SAMUR$))=SAMUR$
180 POKE 53248,X1:POKE 53249,X1:
POKE 53250,X2:POKE 53251,X2
190 FOR C=0 TO 3:POKE 53252+C,X3-2*C:NEXT C
500 DATA 12,6,18,10,2,0,0,0,0,136,140,0,
28,28,28,20,36,36,36,36,36,102,231
510 DATA 0,8,12,20,28,28,12,14,30,54,
114,28,0,0,0,0,0,0,0,0
520 DATA 0,24,60,40,60,60,124,112,124,
232,238,206,124,0,0,68,108,108,108,68,
68,68,102
530 DATA 128,224,192,84,64,64,0,12,0,17,
17,49,0,124,124,56,0,0,0,0
540 DATA 128,140,156,144,156,156,152,188,
190,254,254,189,153,188,60,38,98,66,195,
195
```

לפני שנתקדם הלאה, שים לב לכמה דברים בתוכנה שזה אך הקלדת:

- בשורה מס' 70, הצבנו את הערך 62 לתוך בית 559. כזכור, הערך 62 מורה למחשב כי אתה משתמש בהפרדה אופקית של שורה אחת (בניגוד לערך 46 המורה על שימוש בהפרדה אופקית של שתי שורות)
- באותה שורה, הצבנו את הערך 17 לתוך בית 623. כזכור, ערך זה מורה למחשב להתייחס אל ארבעת הטילים כאל "שחקן חמישי", כלומר צובע את כל הטילים בצבע אחד חדש.
- בשורה מס' 90, הצבנו ערכים לתוך כל הבתים האחראיים לצבעם של השחקנים. בית 711 אחראי, כזכור, לצבעו של "השחקן החמישי" – כל עוד הוגדר ככזה.
- בשורה מס' 190, הגדרנו את מקומו האופקי של השחקן החמישי. מכיוון ש"שחקן" זה בנוי למעשה משילוב של כל הטילים, פרשנו את ארבעת הטילים, זה לצד זה – בקפיצות של שני פיקסלים.

אנימציה

מהירות הטיפול של המחשב במשתנים המחרוזתיים פותחת גם אפשרות להנפיש את השחקנים שלך. היזכר במשחקי המחשב המלהיבים של מחשב האטארי שלך. כמעט ולא תמצא בהם דמויות סטאטיות. ככל שהמשחק משוכלל יותר, הדמויות ינועו אגב הזזת רגליהם או ידיהם, ימצמצו בעיניהם ועוד כהנה וכהנה. אין מניעה שגם אתה תנצל את הידע שרכשת בפרק החמישי ותוסיף אנימציה לשחקנים שלך.

בדוגמה הבאה, נשוב אל ברוס-לי ונוסיף לו אפשרות לבעוט ביריביו. לצורך הלימוד נמחק את הסומו והסמוראי ונתמקד רק בדמותו של ברוס-לי.

■ שנה (והוסף) בתוכנית שהקלדת לעיל את השורות הבאות:

```
30 DIM BRUC1$(30),BRUC2$(30),BRUCB1$(30),
BRUCB2$(30)
60 X=180: X1=180: Y=150
70 GRAPHICS 0: POKE 53277,3: POKE 559,62:
POKE 623,32: POKE 53256,0
120 FOR R=1 TO 23:READ M:
BRUCB1$(R,R)=CHR$(M): NEXT R
130 FOR R=1 TO 20: READ M:
BRUCB2$(R,R)=CHR$(M): NEXT R
140 GOSUB 210
150 IF STRIG(0)=0 THEN GOSUB 170
160 GOTO 150
170 X1=X+4: X=X-4: POKE 53256,1
180 PL0$(Y,Y+LEN(BRUCB1$))=BRUCB1$:
PL1$(Y,Y+LEN(BRUCB2$))=BRUCB2$
190 POKE 53248,X:POKE 53249,X1
200 FOR TIME=1 TO 250:NEXT TIME: X=X+4:
X1=X: POKE 53256,0
210 PL0$(Y,Y+LEN(BRUC1$))=BRUC1$:
PL1$(Y,Y+LEN(BRUC2$))=BRUC2$
220 POKE 53248,X: POKE 53249,X1
230 RETURN
520 DATA 2,1,3,3,1,0,0,12,4,248,252,77,5,
4,12,28,28,0,0,0,0,0,0
530 DATA 0,12,22,26,126,252,254,159,91,
57,17,0,0,0,0,0,0,0,0,0
```

התבונן בבעיטה שנותן ברוס-לי (עם לחיצה על כפתור הג'ויסטיק). מהירות התנועה היא תנאי הכרחי ליצירת האנימציה, ובאמצעות שני צמדים של משתנים מחרוזתיים אכן ניתן במהירות ובקלות רבה להחליף בין מצב א' (ברוס-לי נח) למצב ב' (ברוס-לי בועט). כמובן שאפשר ליצור גם מצבים רבים נוספים, כגון: ברוס-לי רץ, מטפס, נותן אגרוף וכולי.

עיין עתה בשורות 170-200, בתוכנה שהקלדת. שים לב, על מנת ליצור את הבעיטה, נדרשנו לחרוג ממסגרת הרוחב של השחקן (כזכור, בת 8 פיקסלים). על מנת "להרוויח" עוד כמה פיקסלים, ניצלנו את האפשרות למתוח את רוחבו של שחקן, ומתחנו את רוחבו של השחקן השחור (אחד משני השחקנים המרכיבים את דמותו של ברוס-לי).

***

את כל הידע הרב שרכשת ביחידת לימוד זו בנושא גרפיקת השחקנים תוכל עתה לנצל לבניית משחקי פעולה מהירים ומאתגרים, אנימציות מלהיבות ועוד כהנה וכהנה שימושים מועילים.

נאחל לך בהצלחה במלאכת התכנות!

סיכום

לנוחיותך, ריכזנו להלן את כל כתובות הזיכרון החשובות בהפעלתה של גרפיקת השחקנים:

תפקיד	כתובת זיכרון
א. הגדרות יסודיות	
"פתיחת מתג" ה-PMG ערך 1 – פותח לשימוש את השחקנים בלבד. ערך 2 – פותח לשימוש את הטילים בלבד. ערך 3 – פותח לשימוש את השחקנים והטילים.	POKE 53277,
הגדרת סוג ההפרדה האופקית ערך 46 – הפרדה אופקית של שתי שורות. ערך 62 – הפרדה אופקית של שורה אחת.	POKE 559,
בית מצביע המורה למחשב מהיכן לשאוב את נתוני ה-PMG	POKE 54279,
הגדרת התנהגות השחקנים והטילים על המסך: ערך 1 – כל השחקנים בקדמת הגרפיקה, שחקן 0 לפני שחקן 1; השאר לפי הסדר ערך 32 – כל השחקנים בקדמת הגרפיקה, שחקן 1 לפני שחקן 0; השאר לפי הסדר ערך 2 – שחקנים 0,1 בקדמת הגרפיקה; השאר מאחור ערך 4 – כל השחקנים מאחורי הגרפיקה ערך 8 – כל השחקנים מאחורי גרפיקה בצבע 3, אך לפני גרפיקה בצבעים 1,2 ערך 17 – ארבעת הטילים מתנהגים כשחקן חמישי	POKE 623,
ב. מיקומם האופקי של השחקנים	
שחקן מס׳ 0	POKE 53248,
שחקן מס׳ 1	POKE 53249,
שחקן מס׳ 2	POKE 53250,
שחקן מס׳ 3	POKE 53251,
טיל מס׳ 0	POKE 53252,
טיל מס׳ 1	POKE 53253,
טיל מס׳ 2	POKE 53254,
טיל מס׳ 3	POKE 53255,

ג. צבע	
צבע שחקן (+טיל) מס' 0	POKE 704,
צבע שחקן (+טיל) מס' 1	POKE 705,
צבע שחקן (+טיל) מס' 2	POKE 706,
צבע שחקן (+טיל) מס' 3	POKE 707,
צבע "השחקן החמישי"	POKE 711,
ד. ניטור התנגשויות	
1. התנגשויות בין טילים לאלמנטים גרפיים	
טיל מס' 0	PEEK (53248)
טיל מס' 1	PEEK (53249)
טיל מס' 2	PEEK (53250)
טיל מס' 3	PEEK (53251)
2. התנגשויות בין שחקנים לאלמנטים גרפיים	
שחקן מס' 0	PEEK (53252)
שחקן מס' 1	PEEK (53253)
שחקן מס' 2	PEEK (53254)
שחקן מס' 3	PEEK (53255)
3. התנגשויות בין טילים לשחקנים	
טיל מס' 0	PEEK (53256)
טיל מס' 1	PEEK (53257)
טיל מס' 2	PEEK (53258)
טיל מס' 3	PEEK (53259)
4. התנגשויות בין שחקנים לשחקנים אחרים	
שחקן מס' 0	PEEK (53260)
שחקן מס' 1	PEEK (53261)
שחקן מס' 2	PEEK (53262)
שחקן מס' 3	PEEK (53263)
איפוס כל ההתנגשויות	POKE 53278,1
ה. שחקנים מתוחים ומתוחים כפליים	
שחקן מס' 0 (ערך 0 – רגיל, או ערכים 1,3 למתיחה)	POKE 53256,
שחקן מס' 1 (כנ"ל)	POKE 53257,
שחקן מס' 2 (כנ"ל)	POKE 53258,
שחקן מס' 3 (כנ"ל)	POKE 53259,
ערכים 1,3 למתיחת טיל מס' 0 ערכים 4,12 למתיחת טיל מס' 1 ערכים 16,48 למתיחת טיל מס' 3 ערכים 64,192 למתיחת טיל מס' 4	POKE 53260,

תשובות

תשובה 1

256X152=38912

תשובה 2

5X128=640

תשובה 3

1024 הם 256X4 ולכן עליך לפקוד על המחשב:

```
POKE 106,(PEEK(106)-4)
```

תשובה 4

56, 56, 16, 56, 92, 154, 24, 60, 36, 36

תשובה 5

(3X16)+4=52

תשובה 6

```
10 A=PEEK(106)-4: POKE 106,A: GR.0:
POKE 54279,A
20 X=100:Y=50
30 PMBASE=A*256:FOR R=PMBASE+512 TO
PMBASE+640: POKE R,0: NEXT R
40 POKE 53277,3: POKE 559,46
```

```
50 FOR R=PMBASE+512+Y TO PMBASE+512+Y+11:
READ M: POKE R,M: NEXT R
60 POKE 53248,X
70 IF STICK(0)=7 THEN X=X+1:GOTO 60
80 IF STICK(0)=11 THEN X=X-1:GOTO 60
90 IF STICK(0)=13 THEN FOR R=PMBASE+512+Y+11
TO PMBASE+512+Y STEP -1: POKE R+1,PEEK(R):
NEXT R: Y=Y+1
100 IF STICK(0)=14 THEN FOR R=PMBASE+512+Y
TO PMBASE+512+Y+11:POKE R-1,PEEK(R):
NEXT R: Y=Y-1
110 GOTO 70
200 DATA 0,56,56,16,56,92,154,24,60,36,
36,0
```

<u>תשובה 7</u>

0, 60, 126, 219, 126, 36, 36, 66, 129, 0

<u>תשובה 8</u>

```
10 A=PEEK(106)-4: POKE 106,A: GR. 3:
POKE 54279,A
20 XM=100: YM=50: XE=120: YE=50
30 PMBASE=A*256: FOR R=PMBASE+512 TO
PMBASE+768: POKE R,0: NEXT R
40 POKE 53277,3: POKE 559,46
```

```
50 FOR R=PMBASE+512+YM TO PMBASE+512+YM+11:
READ M: POKE R,M: NEXT R
60 FOR R=PMBASE+640+YE TO PMBASE+640+YE+9:
READ M: POKE R,M: NEXT R
70 POKE 53248,XM: POKE 53249,XE
80 POKE 704,132: POKE 705,168
100 DATA 0,56,56,16,56,92,154,24,60,36,
36,0
110 DATA 0,60,126,219,126,36,36,66,129,0
```

<u>תשובה 9</u>

השורות שיש להוסיף/לשנות

```
20 XM=100: YM=50: XE=120: YE=50: XE2=140:
YE2=50: XE3=160: YE3=50
30 PMBASE=A*256: FOR R=PMBASE+512 TO
PMBASE+1024: POKE R,0: NEXT R
63 REST. 110: FOR R=PMBASE+768+YE2 TO
PMBASE+768+YE2+9: READ M: POKE R,M: NEXT R
67 REST. 110: FOR R=PMBASE+896+YE3 TO
PMBASE+896+YE3+9: READ M: POKE R,M:NEXT R
75 POKE 53250,XE2: POKE 53251,XE3
80 POKE 704,132: POKE 705,168: POKE 706,
52: POKE 707,86
```

תשובה 10

טיל 1 יכול לקבל את הערכים: 4, 8, 12

טיל 2 יכול לקבל את הערכים: 16, 32, 48

טיל 3 יכול לקבל את הערכים: 64, 128, 192

תשובה 11

אלו הן השורות שיש להוסיף/לשנות:

```
20 XM=100: YM=100: XE=50: YE=30: YT=97:
YET=33
105 IF (INT(RND(0)*5)=4) AND (ETIL=0)
THEN ETIL=1: XET=XE+3
115 IF ETIL=1 THEN GOSUB 400
400 YET=YET+3: IF YET>125 THEN ETIL=0:
YET=33: RETURN
410 POKE 53253,XET: R=PMBASE+384+YET:
POKE R,0: POKE R+1,0: POKE R+2,0:
POKE R+3,4: POKE R+4,4
420 IF ABS(YET-YM)<4 AND ABS(XET-XM)<4
THEN 450
430 RETURN
450 POKE 53252,0: POKE 53253,0: FOR R=1
TO 255: POKE 704,R: NEXT R: POKE 53248,0
460 ? "GAME OVER!": END
```

תשובה 12

```
90 P=PEEK (53257):IF P=1 THEN POKE 704,52
```

תשובה 13

```
120 P2=PEEK (53249): IF P2<>0 THEN XT=0:
POKE 53278,1
```

תשובה 14

```
10 DIM F1$(1): A1=ADR(F1$)
20 A2=INT(A1/1024)+1: A3=(A2*1024-A1)-1
30 DIM F2$(A3)
40 DIM BUFFER$(384), MSLS$(128), PL0$(128),
PL1$(128), PL2$(128), PL3$(128)
50 PMBASE=ADR(BUFFER$): POKE 54279,
(PMBASE/256)
60 POKE 53277,3: POKE 559,46: POKE 704,132:
POKE 705,168
```

תשובה 15

```
70 BUFFER$(1)=CHR$(0): BUFFER$(384)=CHR$(0):
BUFFER$(2)=BUFFER$
80 MSLS$=BUFFER$: PL0$=BUFFER$:
PL1$=BUFFER$: PL2$=BUFFER$: PL3$=BUFFER$
```

תשובה 16

```
90 DIM GIBOR$(12), ET$(10)
100 FOR R=1 TO 12: READ M:
GIBOR$(R,R)=CHR$(M): NEXT R
110 FOR R=1 TO 10: READ M:
ET$(R,R)=CHR$(M): NEXT R
1000 DATA 0,56,56,16,56,92,154,24,60,
36,36,0
1010 DATA 0,60,126,219,126,36,36,66,
129,0
```

תשובה 17

```
120 XM=100: YM=100: XE=50: YE=30
130 PL0$(YM,YM+11)=GIBOR$
140 PL1$(YE,YE+9)=ET$
150 POKE 53248,XM: POKE 53249,XE
```

תשובה 18

```
200 XM=XM+((PEEK(632)=7)*(XM<212))-
2*((PEEK(632)=11)*(XM>35)): POKE 53248,XM
210 IF (PEEK(632)=14 AND YM>2) THEN YM=YM-1:
PL0$(YM,YM+11)= GIBOR$
220 IF (PEEK(632)=13 AND YM<110) THEN YM=YM+1:
PL0$(YM,YM+11)= GIBOR$
230 GOTO 200
```

יחידת לימוד זאת הנקראת "גרפיקת השחקנים" מתמקדת באלמנטים הגרפיים הייחודיים למחשב האטארי.

ביחידה לימוד זאת נסקור לפניך סט רחב וחדש של יכולות גרפיות שצופן בחובו המחשב ונספק לך כלים יעילים כדי לפתח בכוחות עצמך משחקי פעולה מרהיבים שלא נופלים ברמתם ממשחקי הארקייד.

נכתב ע"י: נדב שטכמן פולישוק